INFO2009-Introduction à l'informatique

Aide à la réussite - Les types abstraits

Bertrand Alexis

Université de Liège

2025

Les Structures de données

Les structures de données possèdent une interface et une implémentation. L'interface reste inchangée et correspond à une sorte de documentation permettant d'utiliser la structure de données. L'implémentation correspond à comment la structure et ses fonctions ont été écrites dans le code.

Les listes

Une liste est une structure de donnée capable de retenir une collection de valeurs. Une liste permet à l'utilisateur d'ajouter un nouvel élément et de retirer un élément librement (peu importe sa place).

Les listes sont la base de structures de données plus complexes. Celles-ci peuvent être implémentées de deux manières :

- Tableau
- Liste chaînée
  - Liste simplement chaînée
  - Liste doublement chaînée

Les piles

Une pile est une structure de données capable de retenir une collection de valeurs. Une pile permet à l'utilisateur d'ajouter un nouvel élément et de retirer un élément en suivant le principe *LIFO* (Last In First Out).

L'interface contient les fonctions suivantes :

- push(p, v) : empile (ajoute au-dessus) la valeur v sur la pile p.
- pop(p) : dépile (enlève au-dessus) une valeur de p et retourne cette valeur.
- top(p) : retourne la valeur au-dessus de la pile.
- size(p) : retourne la taille de p.
- is_empty(p) : retourne une valeur booléenne indiquant si la pile est vide ou pas.

Les files

Une file est une structure de données capable de retenir une collection de valeurs. Une file permet à l'utilisateur d'ajouter un nouvel élément et de retirer un élément en suivant le principe *FIFO* (First In First Out).

L'interface contient les fonctions suivantes :

- send(f, v) : ajoute la valeur v à la file f.
- receive(f, v) : extrait la valeur la plus ancienne de la file et la retourne.
- size(f) : retourne la taille de f.
- is_empty(f) : retourne une valeur booléenne indiquant si la pile est vide ou pas.

D'autres types abstraits

- Arbre binaire
- Graphe
- File de priorité
- Dictionnaire
- ...

Exercices

Un navigateur `WWW` enregistre l'historique d'une session en mémorisant les pages Web qui ont été visitées, ainsi que le moment où elles ont été visitées. Chaque page Web est identifiée par une chaîne de caractères appelée URL (*Uniform Resource Locator*), par exemple `"https://www.uliege.be"`. La date et l'heure d'une visite sont représentées par un seul nombre entier de type `unsigned long` correspondant au nombre de secondes écoulées depuis une date de référence fixée au premier janvier 1970 à 0h00. De plus, pour chaque page faisant partie de l'historique, on conserve un moyen d'accès à la page précédente et à la page suivante (par rapport à l'ordre chronologique où elles ont été visitées), de façon à pouvoir naviguer en arrière ou en avant.

Exercices

1. Écrire un fragment de code définissant un type structuré capable de représenter une page Web contenue dans un historique. Ce type structuré doit comprendre :
   - un pointeur vers une chaine de caractères donnant l'URL de la page,
   - la représentation de la date et l'heure de visite de la page,
   - deux pointeurs vers la page précédente et la page suivante dans l'historique. (Si l'une de ces pages n'existe pas, le pointeur concerné est alors vide.)

Exercices

2. Écrire une fonction prenant en arguments un tableau `url` de chaînes de caractères constituant les URLs de pages visitées, dans l'ordre chronologique de leur visite, un tableau `temps` d'entiers contenant les dates et heures de visite de ces pages (en d'autres termes, `temps[i]` contient le moment où la page `url[i]` a été visitée), et la taille n commune de ces deux tableaux, supposée non nulle. On demande que la fonction crée un nouvel historique de navigation pour les n pages fournies, en allouant dynamiquement la mémoire nécessaire, et retourne un pointeur vers la représentation de la première page faisant partie de cet historique.

Exercices

3. Écrire une fonction prenant en argument un pointeur vers n'importe quelle page d'un historique de navigation, et qui retourne le nombre de secondes écoulées entre les moments où la première et la dernière page de cet historique ont été visitées.
4. Écrire une procédure prenant en argument un pointeur vers n'importe quelle page d'un historique de navigation créé au point (2) et libérer toute la mémoire allouée.

Exercices

5. Implémenter une pile à l'aide d'un vecteur et à l'aide d'une liste.